What Is Claimed Is:

1. A method for generating code to perform anticipatory prefetching for data references, comprising:

receiving code to be executed on a computer system;

analyzing the code to identify data references to be prefetched, wherein analyzing the code involves,

performing a first marking phase in which only data references located in blocks that are certain to execute are considered in determining which data references are covered by preceding data references, and

performing a second marking phase in which data references that are located in blocks that are not certain to execute are considered; and

inserting prefetch instructions into the code in advance of the identified data references.

2. The method of claim 1, further comprising:

profiling execution of the code to produce profiling results; and

using the profiling results to determine whether a given block of instructions is executed frequently enough to perform the second marking phase on the given block of instructions.

3. The method of claim 2, wherein determining whether the given block of instructions is executed frequently enough to perform the second marking phase involves comparing a frequency of execution for the given block from the profiling results with a threshold value indicating a minimum frequency of execution to be considered in the second marking phase.

4. The method of claim 1, wherein analyzing the code involves:

identifying loop bodies within the code; and

identifying data references to be prefetched from within the loop bodies.

5. The method of claim 4, wherein if there exists a nested loop within the code, analyzing the code involves:

examining an innermost loop in the nested loop; and

examining a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

6. The method of claim 4, wherein analyzing the code to identify data references to be prefetched involves examining a pattern of data references over multiple loop iterations.

7. The method of claim 1, wherein analyzing the code involves analyzing the code within a compiler.

8. A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for generating code to perform anticipatory prefetching for data references, the method comprising:

receiving code to be executed on a computer system;

analyzing the code to identify data references to be prefetched, wherein analyzing the code involves,

performing a first marking phase in which only data references located in blocks that are certain to execute are considered in determining which data references are covered by preceding data references, and

performing a second marking phase in which data references that are located in blocks that are not certain to execute are considered; and

inserting prefetch instructions into the code in advance of the identified data references.

9. The computer-readable storage medium of claim 8, wherein the method further comprises:

profiling execution of the code to produce profiling results; and

using the profiling results to determine whether a given block of instructions is executed frequently enough to perform the second marking phase on the given block of instructions.

10. The computer-readable storage medium of claim 9, wherein determining whether the given block of instructions is executed frequently enough to perform the second marking phase involves comparing a frequency of execution for the given block from the profiling results with a threshold value indicating a minimum frequency of execution to be considered in the second marking phase.

11. The computer-readable storage medium of claim 8, wherein analyzing the code involves:

identifying loop bodies within the code; and

identifying data references to be prefetched from within the loop bodies.

12. The computer-readable storage medium of claim 11, wherein if there exists a nested loop within the code, analyzing the code involves:

examining an innermost loop in the nested loop; and

examining a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

13. The computer-readable storage medium of claim 11, wherein analyzing the code to identify data references to be prefetched involves examining a pattern of data references over multiple loop iterations.

14. The computer-readable storage medium of claim 11, wherein analyzing the code involves analyzing the code within a compiler.

15. An apparatus that generates code to perform anticipatory prefetching for data references, comprising:

a receiving mechanism that is configured to receive code to be executed on a computer system;

an analysis mechanism that is configured to analyze the code to identify data references to be prefetched, wherein the analysis mechanism is configured to,

perform a first marking phase in which only data references located in blocks that are certain to execute are considered in determining which data references are covered by preceding data references, and to

perform a second marking phase in which data references that are located in blocks that are not certain to execute are considered; and an insertion mechanism that is configured to insert prefetch instructions into the code in advance of the identified data references.

16. The apparatus of claim 15, further comprising a profiling mechanism that is configured to profile execution of the code to produce profiling results;

wherein the analysis mechanism is configured to use the profiling results to determine whether a given block of instructions is executed frequently enough to perform the second marking phase on the given block of instructions.

17. The apparatus of claim 16, wherein the analysis mechanism is configured to compare a frequency of execution for the given block from the profiling results with a threshold value indicating a minimum frequency of execution to be considered in the second marking phase.

18. The apparatus of claim 15, wherein the analysis mechanism is configured to:

identify loop bodies within the code; and to

identify data references to be prefetched from within the loop bodies.

19. The apparatus of claim 18, wherein if there exists a nested loop within the code, the analysis mechanism is configured to:

examine an innermost loop in the nested loop; and to

examine a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

20. The apparatus of claim 18, wherein the analysis mechanism is configured to examine a pattern of data references over multiple loop iterations.

21. The apparatus of claim 15, wherein the apparatus resides within a compiler.

22. A method for generating code to perform anticipatory prefetching for data references, comprising:

receiving code to be executed on a computer system;

analyzing the code to identify data references to be prefetched, wherein analyzing the code involves,

examining an array reference made through an array subscript,

determining a function for the array subscript in terms of a loop index,

using the function to calculate a difference between array indexes for consecutive loop iterations, and

considering the array reference as a candidate for prefetching if the difference between array indexes for consecutive loop iterations is a constant value; and

inserting prefetch instructions into the code in advance of the identified data references.

23. The method of claim 22, wherein determining the function for the array subscript in terms of a loop index involves chasing down data dependencies associated with the array subscript if such data dependencies exist.

24. The method of claim 22, wherein the array reference is considered a candidate for prefetching if the difference between array indexes is a constant value for some but not all consecutive loop iterations.

25. The method of claim 24, wherein the array reference is considered a candidate for prefetching if the difference between array indexes depends on a modulo operator that causes the difference between array indexes to occasionally vary from the constant value.

26. The method of claim 22, wherein analyzing the code involves:

identifying loop bodies within the code; and

identifying data references to be prefetched from within the loop bodies.

27. The method of claim 26, wherein if there exists a nested loop within the code, analyzing the code involves:

examining an innermost loop in the nested loop; and

examining a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

28. The method of claim 26, wherein analyzing the code involves examining a pattern of data references over multiple loop iterations.

29. The method of claim 22, wherein analyzing the code involves analyzing the code within a compiler.

30. A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for generating code to perform anticipatory prefetching for data references, the method comprising:

receiving code to be executed on a computer system;

analyzing the code to identify data references to be prefetched, wherein analyzing the code involves,

examining an array reference made through an array subscript,

determining a function for the array subscript in terms of a loop index,

using the function to calculate a difference between array indexes for consecutive loop iterations, and

considering the array reference as a candidate for prefetching if the difference between array indexes for consecutive loop iterations is a constant value; and

inserting prefetch instructions into the code in advance of the identified data references.

31. The computer-readable storage medium of claim 30, wherein determining the function for the array subscript in terms of a loop index involves chasing down data dependencies associated with the array subscript if such data dependencies exist.

32. The computer-readable storage medium of claim 30, wherein the array reference is considered a candidate for prefetching if the difference between array indexes is a constant value for some but not all consecutive loop iterations.

33. The computer-readable storage medium of claim 32, wherein the array reference is considered a candidate for prefetching if the difference between array indexes depends on a modulo operator that causes the difference between array indexes to occasionally vary from the constant value.

34. The computer-readable storage medium of claim 30, wherein analyzing the code involves:

identifying loop bodies within the code; and

identifying data references to be prefetched from within the loop bodies.

35. The computer-readable storage medium of claim 34, wherein if there exists a nested loop within the code, analyzing the code involves:

examining an innermost loop in the nested loop; and

examining a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

36. The computer-readable storage medium of claim 34, wherein analyzing the code involves examining a pattern of data references over multiple loop iterations.

37. The computer-readable storage medium of claim 30, wherein analyzing the code involves analyzing the code within a compiler.

38. An apparatus that generates code to perform anticipatory prefetching for data references, comprising:

a receiving mechanism that is configured to receive code to be executed on a computer system;

an analysis mechanism that is configured to analyze the code to identify data references to be prefetched, wherein the analysis mechanism is configured to,

examine an array reference made through an array subscript,

determine a function for the array subscript in terms of a loop index,

use the function to calculate a difference between array indexes for consecutive loop iterations, and to

consider the array reference as a candidate for prefetching if the difference between array indexes for consecutive loop iterations is a constant value; and

an insertion mechanism that is configured to insert prefetch instructions into the code in advance of the identified data references.

39. The apparatus of claim 38, wherein while determining the function for the array subscript in terms of a loop index, the analysis mechanism is configured to chase down data dependencies associated with the array subscript if such data dependencies exist.

40. The apparatus of claim 38, wherein the analysis mechanism is configured to consider the array reference as a candidate for prefetching if the difference between array indexes is a constant value for some but not all consecutive loop iterations.

41. The apparatus of claim 40, wherein the analysis mechanism is configured to consider the array reference as a candidate for prefetching if the difference between array indexes depends on a modulo operator that causes the difference between array indexes to occasionally vary from the constant value.

42. The apparatus of claim 38, wherein the analysis mechanism is configured to:

identify loop bodies within the code; and to

identify data references to be prefetched from within the loop bodies.

43. The apparatus of claim 42, wherein if there exists a nested loop within the code, the analysis mechanism is configured to:

examine an innermost loop in the nested loop; and to

examine a loop outside the innermost loop if the innermost loop is smaller than a minimum size or is executed fewer than a minimum number of iterations.

44. The apparatus of claim 42, wherein the analysis mechanism is configured to analyze a pattern of data references over multiple loop iterations.

45. The apparatus of claim 38, wherein the apparatus resides within a compiler.